U.S.P.S. Express Mail Mailing Label No.: EV 303 831 513 US

Date of Deposit: January 2, 2004

Attorney Docket No. 14230US03

SYSTEM AND METHOD FOR HANDLING TRANSPORT PROTOCOL SEGMENTS

CROSS-REFERENCE TO RELATED APPLICATIONS

[01] This application makes reference to, claims priority to and claims benefit from United States Provisional Patent Application Serial No. 60/437,887, entitled "Header Alignment and Complete PDU" and filed on January 2, 2003; and United States Provisional Patent Application Serial No. 60/456,322, entitled "System and Method for Handling Transport Protocol Segments" and filed on March 20, 2003.

INCORPORATION BY REFERENCE

[02] The above-referenced United States patent applications are hereby incorporated herein by reference in their entirety.

FEDERALLY SPONSORED RESEARCH OR DEVELOPMENT

[03] [Not Applicable]

[MICROFICHE/COPYRIGHT REFERENCE]

[04] [Not Applicable]

BACKGROUND OF THE INVENTION

[05] FIG. 1 shows a conventional byte stream in accordance with a transmission control protocol (TCP). Three segments (i.e., TCP Seg. X-1, TCP Seg. X and TCP Seg. X+1) of the byte stream are illustrated. There is no guaranteed relationship between an upper layer protocol data unit (ULPDU) and TCP segments boundaries. As a result, a ULPDU may start or end in the middle of the TCP Segment. For example, two ULPDUs (e.g., ULPDU Y and

ULPDU Y+1) are each carried by two TCP segments. A ULPDU may also be carried by more than two TCP segments.

[06] In conventional systems, by carrying each ULPDU over two or more TCP segments, a network interface card (NIC) of a receiver may have to perform excessive computations and operations that can hamper NIC performance in very high speed networks such as, for example, networks with bandwidths exceeding one gigabit per second (Gbps). The receiver may have difficulty in determining the beginning of each ULPDU in, for example, a seemingly endless TCP byte stream. In addition, the receiver may need to process the IP datagram as well as TCP segments, to determine the upper layer protocol (ULP) boundaries and to perform ULP CRC before the ULPDU header placement information can be trusted. Determining the beginning of each ULPDU and trusting the ULPDU header placement information are but a few of the obstacles in developing, for example, a NIC in which the NIC, with minimum buffering or no buffering, may directly place the ULPDU data into a designated host buffer location.

[07] Another obstacle to developing, for example, a NIC that can place ULPDUs into host memory may be the buffer memory requirements of the NIC. Since the ULPDU cannot be placed until the entire ULPDU has been buffered and respective control information analyzed, buffers are needed to accommodate, for example, out-of-order TCP segments that may disrupt the flow of ULPDUs. A TCP receiver may allocate buffers based upon, for example, a bandwidth-delay product. Thus, the buffer memory size may scale linearly with network speed. For example, an approximately tenfold increase in network speed may necessitate an approximately tenfold increase in buffer memory. This causes the total cost of a NIC for high speed network to increase to a level that makes it impractical for wide deployment. In addition, the memory may be managed on a per connection basis. Each receiver connection may require its own buffers since each ULPDU may be carried by a plurality of TCP segments. Such buffering requirements can only be accentuated as network speeds and the number of connections increase.

[08] Further limitations and disadvantages of conventional and traditional approaches will become apparent to one of ordinary skill in the art through comparison of such systems with

some aspects of the present invention as set forth in the remainder of the present application with reference to the drawings.

BRIEF SUMMARY OF THE INVENTION

[09] Aspects of the present invention may be found in, for example, systems and methods that handle transport protocol segments (TPSes). In one embodiment, the present invention may provide a system that includes, for example, a receiver that may receive an incoming TPS. The incoming TPS may include, for example, an aligned upper layer protocol (ULP) header and a complete ULP data unit (ULPDU). The receiver may directly place the complete ULPDU into a host memory.

[10] In another embodiment, the present invention may provide a system that handles TPSes. The system may include, for example, a sender that sends a TPS. The sent TPS may include, for example, an aligned ULP header and one or more complete ULPDUs.

[11] In another embodiment, the present invention may provide a method that handles TPSes. The method may include, for example, one or more of the following: aligning an FPDU header in a known position in a TPS with respect to a TPS header; and placing a complete FPDU in the TPS.

[12] In yet another embodiment, the present invention may provide a method that handles TPSes. The method may include, for example, receiving an incoming TPS. The TPS may include, for example, a complete FPDU and an FPDU header in a known position with respect to a TPS header.

[13] In yet another embodiment, the present invention may provide a system that handles TPSes. The system may include, for example, a receiver including a direct memory access (DMA) engine. The receiver may receive an incoming TPS that includes an aligned ULP header and a complete ULPDU. The receiver may program the DMA engine once to place the complete ULPDU into a host memory.

[14] In another embodiment, the present invention may provide a method that handles TPSes. The method may include one or more of the following: receiving an incoming TPS, the TPS comprising a complete FPDU and an FPDU header in a known position with respect to a TPS header; performing layer 2 (L2) processing on the incoming TPS; performing layer

3 (L3) processing on the incoming TPS; performing layer 4 (L4) processing on the incoming TPS; and performing ULP processing on the incoming TPS. The L2 processing, the L3 processing, the L4 processing and the ULP processing of the incoming TPS may be performed in any order.

[15] These and other features and advantages of the present invention may be appreciated from a review of the following detailed description of the present invention, along with the accompanying figures in which like reference numerals refer to like parts throughout.

BRIEF DESCRIPTION OF THE DRAWINGS

[16] FIG. 1 shows an upper layer protocol data unit (ULPDU) carried by a plurality of transmission control protocol (TCP) segments.

[17] FIG. 2 shows an embodiment of a system that handles framing protocol data units (FPDUs) carried by TCP segments according to the present invention.

[18] FIG. 3 shows an embodiment of a system that handles TCP frames in a flow-through manner according to the present invention.

[19] FIG. 4 shows another embodiment of a system that handles TCP frames in a flow-through manner according to the present invention.

[20] FIG. 5 shows an embodiment of an FPDU carried by a respective TCP segment according to the present invention.

[21] FIGS. 6A-B show an embodiment of a method that processes FPDU according to the present invention.

[22] FIG. 7 shows an embodiment of a network interface card (NIC) according to the present invention.

[23] FIG. 8 shows an embodiment of a NIC according to the present invention.

DETAILED DESCRIPTION OF THE INVENTION

[24] If an upper layer protocol data unit (ULPDU) is not aligned within a transport segment, then the ULPDU may be carried by two or more transport protocol segments (e.g., two or more transmission control protocol (TCP) segments) and a receiver may perform layer 2 (L2) processing on incoming frames. If an Internet protocol (IP) datagram is not an IP fragment, then the L2 frame may include a complete IP datagram and layer 3 (L3) processing may be performed on the IP datagram. If an IP fragment is present in the L2 frame, then IP fragments may be reassembled in a local buffer before continuing with the processing. Some protocols (e.g., IP security (IPsec)) may include, for example, a header between an L3 header and an L4 header (or between other headers) and may be dealt with as is known in the art; however, such considerations will not be discussed further herein to simplify the discussion. Subsequently, layer 4 (L4) processing may commence including, for example, TCP processing and performing header/checksum checks. The L4 segment (e.g., a TCP segment, a stream control transmission protocol (SCTP) segment or other transport layer segment) may be classified, for example, to determine a "flow". For TCP/IP traffic this may be done using a 5-tuple including, for example, IP source information, IP destination information, TCP source port information, TCP destination port information and protocol information). State information for L3/L4 and any upper layer protocol (ULP) may then be obtained for the flow such as for a TCP connection. If the receiver has kept state for the upper layer protocol (ULP), then the boundaries of the ULPDU message may be determined. Based on the ULPDU boundaries and ULP header information, the payload boundaries along with the requested action (e.g., determining payload placement information) may be ascertained.

[25] If a TCP segment is not received in order (e.g., the TCP segment is an out-of-order TCP segment) and if a ULPDU is not aligned within the TCP segment, then the receiver may buffer the data until a complete ULPDU is received. In most cases, it may be difficult to determine the ULPDU boundaries inside out-of-order TCP segments. The receiver may not be able to immediately calculate a ULP cyclical redundancy checking (CRC) (herein ULP CRC or CRC), if used, or may not be able to immediately place the ULP payload in a host

buffer. The receiver may take action, for example, after the transport layer segment has been received in order, fully reassembled and tested for transport layer integrity and the ULPDU boundaries have been found. The receiver may buffer out-of-order transport protocol segments or may drop them. Once a complete ULPDU has been received, a process similar to receiving in-order TCP segments may be implemented.

[26] If a TCP segment is received in order (or re-ordered by the receiver) and if a ULPDU is not aligned within the TCP segment, then receiver-managed ULP state information may be used to calculate where, in TCP segment X, for example, the previous ULPDU (e.g., ULPDU Y) ends. This operation may include, for example, the step of subtracting the number of bytes of ULPDU Y in TCP segment X-1 from the ULPDU Y length field stored by the receiver as part of the ULP state. Based on this calculation, the receiver may calculate the number of remaining bytes in TCP segment X that are part of ULPDU Y. If the ULP (e.g., an Internet small computer system interface (iSCSI) protocol, an iSCSI extensions for remote direct memory access (RDMA) (iSER) protocol or other RDMA-over-TCP protocols) employs a data integrity mechanism such as, for example, CRC, then the receiver may calculate the CRC and may check the calculated CRC against the CRC value received. The ULP may have respective CRCs for the ULPDU header and for the data or a single CRC to cover both the ULP header and data. If steering/placement information is included within the ULPDU header and if the ULPDU header has separate CRCs, then data placement may commence once the CRCs are confirmed to be error free. If a single CRC is employed for the whole ULPDU, then CRC across the ULPDU may be computed and checked before placement may begin.

[27] Two embodiments for employing CRC on the receiver are discussed although the present invention may include other embodiments for employing CRC (e.g., conventional CRC). FIG. 7 shows an embodiment of an offload network interface card (NIC) operation for ULP with CRC. The NIC may be, for example, a non-flow-through NIC. In a first method, the CRC may be performed before or as the data is stored in the local buffer. The partial ULPDU CRC results, if applicable, may also be stored in the local buffer or elsewhere such as, for example, in a context memory, a host memory or on-chip. In a second method,

the CRC may be performed as data is moved from local buffer to host buffer. For the first method and the second method, the CRC may be performed by one or more of the blocks illustrated in FIG. 7.

[28] An embodiment of the processing of the ULPDU CRC for reception of a transport protocol segment (e.g., TCP segment X) with an unaligned ULPDU is described. If the first method is used and if the ULPDU is not entirely carried by the transport protocol segment (e.g., TCP segment X), then a partial CRC may have been computed when TCP segment X-1 was received. The stored partial CRC for ULPDU Y, which is a result, for example, of the bytes included in TCP segment X-1, may be fetched and loaded into a CRC circuit that is adapted to continue the CRC calculation starting from the partial CRC, instead of starting from a CRC initialization constant. The receiver may determine the number of bytes within TCP segment X that belong to ULPDU Y. The remaining bytes of ULPDU Y inside TCP segment X may be moved through the CRC machine using the services of a direct memory access (DMA) device, for example. If TCP segment X is received out of order, then the boundaries of ULPDU Y may be difficult to determine and therefore, in some cases, the CRC circuit might not be notified as to a start point or a stop point within TCP Segment X. In such a case, once TCP Segment X is in order, the NIC may determine the boundaries of ULPDU Y. The NIC may re-read the payload from the local memory to the CRC circuit to perform this task. The CRC calculation for ULPDU Y may be checked. If there are no CRC errors, then placement may be allowed. The receiver may then re-arm the DMA and the CRC mechanisms to calculate the partial CRC for ULPDU Y+1 and to store it.

[29] If the second method is used, then data may be placed in the host buffer in parallel with performing CRC. Data may be initially stored in the local memory buffer. Once the ULPDU boundaries are determined, the receiver may add state to keep track of the end of ULPDU Y which may also mark the beginning of ULPDU Y+1. For the ULPDU Y header, the CRC may be checked when TCP segment X-1 is received or when the whole ULPDU is assembled, for example, when TCP segment X is received. If the header CRC is error free, then the steering information contained therein may be trusted. The ULPDU Y data bytes may then be placed in the host buffer. If the ULP uses just a single CRC for both the header

and the data, then the CRC of the whole ULPDU is calculated before any placement may begin. Checking for errors in the CRC for the placement information included within the header before placement commences may safeguard against, for example, data being placed in the wrong host buffer location. For the first method and the second method, if a CRC error is detected, then the ULP may recover by itself.

[30] With regard to moving the data to the host buffer, in some examples (e.g., with unaligned ULPDUs), the two portions of the ULP payload for ULPDU Y may be located in at least two separate local buffers as they were received in at least two separate frames. With regard to the first portion of the ULPDU, the local buffer for the first portion of ULPDU Y (e.g., the portion of ULPDU Y in TCP segment X-1) may be located and the DMA device may be programmed (e.g., *CopyData*(local buffer of first portion of ULPDU Y, host buffer address, length)). The host buffer address for the second portion of ULPDU Y (e.g., the portion of ULPDU Y in TCP segment X) may be computed and the DMA device may be programmed (e.g., *CopyData*(local buffer of second portion of ULPDU Y, host buffer address for second portion, length)).

[31] With respect to moving, for example, ULPDU Y+1 to the host buffer, the receiver may continue with reception of TCP segment X, as it manages the first portion of ULPDU Y+1, which was in TCP segment X. The TCP sequence for the beginning of ULPDU Y+1 may be calculated and the byte offset into the segment may be derived therefrom. The DMA device may be programmed to move available data of ULPDU Y+1 in TCP segment X through the CRC machine, if the first method is used. The partial CRC results may be stored for ULPDU Y+1. The local buffer address of the first portion of ULPDU Y+1 (e.g., the portion of ULPDU Y+1 in TCP segment X) may be stored. If the ULP has a separate CRC for the header and if the header has been fully received and has been found to be error free, then the first ULP payload portion of ULPDU Y+1 may be stored in the host buffer. This may assume that TCP processing has been completed with no errors and that the TCP segment X is in order. If the ULP uses a single CRC to cover both the header and the data, then any placement in the host buffer or any other action with respect to ULPDU Y+1 might be delayed until ULPDU Y+1 has been received in its entirety.

[32] If the ULPDU is aligned (e.g., marker aligned, offset aligned or using other alignment arrangements) with respect to the transport protocol segment, then the operation of the NIC illustrated in FIG. 7 further simplifies. Since the ULPDU is aligned within the transport protocol segment, the boundaries of the ULPDU may be easily discernable with little or no calculation. Since a complete ULPDU is present in each transport protocol segment, the receiver need not store partial portions of the ULPDU or store partial CRC calculations. Thus, whether using the first method or the second method, the CRC machine might only be used once. In fact, after successfully checking the CRC, the receiver might only program the DMA once to place the ULPDU into, for example, the host memory. Furthermore, since a complete ULPDU is present in the transport protocol segment, the ULPDU may include enough information to program the DMA regardless of whether the transport protocol segment is in order or out of order. However, if the checking of the ULP CRC reveals an error, then data may need buffering as this may be an indication of, for example, unaligned ULPDUs or transport errors. The processing of unaligned ULPDUs in a non-flow-through NIC has already been described.

[33] FIG. 8 shows an embodiment of a flow-through offload NIC operation for ULP with CRC according to the present invention. The CRC machine may be in one or more of the blocks illustrated in FIG. 8.

[34] With respect to FIG. 8, if the ULPDU is unaligned, then, for in-order TCP segments, the flow-through NIC operates similarly to the non-flow-through NIC. For out-of-order TCP segments, the flow-through NIC may buffer the out-of-order TCP segments, for example, in the buffer of the TOE/ULP block, drop the out-of-order TCP segments or pass the out-of-order TCP segments to a host software agent for processing along with the partial information it has accumulated such as, for example, a partial ULPDU Y, a partial CRC, etc. When the transport order is restored or based on other criteria, the host software may pass back to the NIC the parameters of the ULP such as, for example, start boundaries, a partial CRC, placement information obtained from its header, etc. The flow-through NIC may then commence processing of the ULPDUs.

[35] If the ULPDU is aligned with the transport protocol segment, then the operation of the flow-through NIC illustrated in FIG. 8 further simplifies. The flow-through NIC may perform the ULP boundary calculation, the CRC checking and the DMA configuration once for an aligned ULPDU instead of multiple times for an unaligned ULPDU. The computation of the ULPDU boundaries may be further simplified since the ULPDU boundaries are aligned within the transport protocol segment. Alignment may also simplify the handling (e.g., computing ULP boundaries, checking CRC and placing in a host buffer) of out-of-order transport protocol segments carrying the aligned ULPDUs.

[36] FIG. 8 shows an embodiment of a flow-through offload NIC operation for ULP with CRC according to the present invention. According to some embodiments of the present invention, the boundaries of the ULPDUs may be defined (e.g., easily determined with respect to) the boundaries of the transport protocol segments. The ULPDU boundaries may be determined for each in-order or out-of-order transport protocol segment. According to various embodiments of the present invention, the CRC may be performed on the whole ULPDU or the whole transport protocol segment payload. Partial CRC results and the storing of partial CRC results may thus be avoided.

[37] In other embodiments of the present invention, the flow-through NIC may place, on the fly, the payload of every transport protocol segment in the host memory, instead of storing the data in a local memory and then forwarding the data to the host memory. In some embodiments according to the present invention, data may be placed in the host memory after ULP CRC or ULP CRCs are calculated and checked, thereby guaranteeing that ULP steering/placement information and the data are intact.

[38] FIG. 2 shows an embodiment of a system that handles ULPDUs such as, for example, framing protocol data units (FPDUs) carried by transport protocol segments such as, for example, TCP segments according to the present invention. A sender system 10 (e.g., a client) may be coupled to a receiver system 30 (e.g., a server) via a network 20 such as, for example, the Internet. One or more TCP connections may be set up between the sender system 10 and the receiver system 30.

[39] FIG. 3 shows an embodiment of a system that handles TCP frames in a flow-through manner according to the present invention. The system may be part of, for example, the sender system 10 and/or the receiver system 30. The system may include, for example, a central processing unit (CPU) 40, a memory controller 50, a host memory 60, a host interface 70, network subsystem 80 and an Ethernet 90. The network subsystem 80 may be, for example, a NIC. The network subsystem 80 may include, for example, a TCP-enabled Ethernet Controller (TEEC) or a TCP offload engine (TOE). The network subsystem 80 may include, for example, a DMA engine and a CRC machine. The DMA engine and the CRC machine may be part of, for example, the TEEC or the TOE. The host interface 70 may be, for example, a peripheral component interconnect (PCI) or another type of bus. The memory controller 50 may be coupled to the CPU 40, to the host memory 60 and to the host interface 70. The host interface 70 may be coupled to the network subsystem 80.

[40] FIG. 4 shows another embodiment of a system that handles TCP frames in a flow-through manner according to the present invention. The system may include, for example, the CPU 40, the host memory 60 and a chip set 100. The chip set 100 may include, for example, the network subsystem 80. The chip set 100 may be coupled to the CPU 40, to the host memory 60 and to the Ethernet 90. The network subsystem 80 of the chip set 100 may be coupled to the Ethernet 90. The network subsystem 80 may include, for example, the TEEC or the TOE which may be coupled to the Ethernet 90. The network subsystem 80 or the chip set 100 may include, for example, a DMA engine and a CRC machine. The DMA engine and the CRC machine may be part of, for example, the TEEC or the TOE. A dedicated memory may be part of and/or coupled to the chip set 100 and may provide buffers for context or data.

[41] Although illustrated, for example, as a CPU and an Ethernet, the present invention need not be so limited to such exemplary examples and may employ, for example, any type of processor and any type of data link layer or physical media, respectively. Accordingly, although illustrated as coupled to the Ethernet 90, the network subsystem 80 may be adapted for any type of data link layer or physical media. Furthermore, the present invention also

contemplates different degrees of integration and separation between the components illustrated in or described with respect to FIGS. 3 and 4.

[42] In operation according to one embodiment of the present invention, the sender 10 may create TCP segments that include, for example, one or more complete FPDUs. The particular length of the FPDUs and the TCP segments may be subject to ULP or network constraints and considerations. In one embodiment, the sender 10 may be an MPA-aware-TCP sender that encapsulates at least one complete FPDU in each TCP segment. An FPDU may be a unit of data created by a ULP using a marker-based ULPDU aligned (MPA) framing protocol. Examples of MPA framing protocols may be found in, for example, United States Patent Application Serial No. 10/230,643, entitled "System and Method for Identifying Upper Layer Protocol Message Boundaries" and filed on August 29, 2002. The above-referenced United States patent application is hereby incorporated herein by reference in its entirety. Other examples of the MPA framing protocols may be found, for example, in conventional MPA framing protocols. An FPDU according to a particular MPA framing protocol may include, for example, an MPA length, an MPA payload, an MPA CRC and, optionally, one or more markers as appropriate.

[43] The TCP segments may be transmitted to the receiver system 30 via, for example, the network 20. The network subsystem 80 may receive the TCP segments via, for example, the Ethernet 90. The network subsystem 80 may receive the TCP segments in order or out of order and may process the TCP segments in a flow-through manner. The network subsystem 80 may determine the boundaries of each FPDU and locate the control information and data information corresponding to each FPDU. The network subsystem 80 may then process the respective control information in order to place the data information directly inside the host memory 60. The network subsystem 80 may employ, for example, a TEEC or a TOE adapted to facilitate the placement of the data contained in the TCP segment into, for example, a temporary buffer, a ULP buffer or an application buffer residing in the host memory 60. For directly placing the data into the host memory 60, the network subsystem 80 may include, for example, a DMA engine. The network subsystem 80 may place the ULP data at a particular memory location, for example, in a ULP buffer residing in the host

memory 60. Accordingly, whether the TCP segment is in order or out of order, the network subsystem 80 may copy the data, for example, from the Ethernet 90 to, for example, a determined buffer location of the ULP buffer residing in the host memory 60.

[44] FIG. 5 shows an embodiment of an FPDU carried by a respective TCP segment according to the present invention. The present invention also contemplates that each TCP segment may carry more than one FPDU. In some embodiments, the present invention may provide that a TCP segment may carry one or more complete FPDUs. In some embodiments, the FPDU may follow immediately after the TCP header. In other embodiments, the FPDU may follow the TCP header after a preset number of bytes. In yet another embodiment, the FPDU may follow the TCP header after a particular number of bytes. The particular number of bytes may be indicated by a field in a known location in the TCP segment or in the TCP byte stream.

[45] FIGS. 6A-B show an embodiment of a method that processes FPDU according to the present invention. In step 120, the network subsystem 80 performs L2 processing on an incoming frame from, for example, the network 20. Assuming that the IP datagram is not an IP fragment (i.e., the L2 frame contains one complete IP datagram), in step 130, the network subsystem may perform L3 processing on the IP datagram. If an IP fragment is present in the L2 frame, then IP fragments must first be reassembled in a local buffer before processing may continue. In step 140, the network subsystem 80 may perform L4 processing including, for example, TCP processing, header checks and checksum checks. In query 150, the network subsystem 80 may check for header alignment. In one embodiment, header alignment may be determined by analyzing the marker in the TCP segment according to, for example, an MPA framing protocol.

[46] If the header is not aligned, then, in step 160, network subsystem 80 or other components of the receiver 30 may perform a processing method for unaligned FPDUs. In one embodiment, the process may be similar to the method that processes unaligned ULPDUs with some differences. For example, under a particular MPA framing protocol, information included in, for example, an MPA length field and one or more MPA markers may be used to locate a particular MPA header and to determine FPDU boundaries. The

MPA header (or a ULP header it carries) may include, for example, information relating to a particular memory location (e.g., a memory address) in the host memory 60 in which data of the FPDU may be placed. In some embodiments according to the present invention, if MPA is not used and if the ULPDU is not aligned, then the NIC may perform additional operations as discussed above with respect to non-aligned ULPDUs

[47] If the header is aligned, then, in step 170, the boundaries of the FPDU including, for example, the location of the FPDU header and the FPDU payload may be determined. The FPDU length information may be obtained from, for example, the FPDU header. Step 170 may be performed whether the TCP segment is an in-order TCP segment or an out-of-order TCP segment. In step 180, a DMA engine may be programmed to move the FPDU data through a CRC machine. In step 190, the CRC calculation for the FPDU may be checked for errors. If the CRC check reveals an error, then, in step 210, the FPDU may be locally dropped or the ULP may initiate recovery. If the CRC check does not reveal an error, then, in step 220, the DMA engine may be programmed to copy data (e.g., *CopyData*(TCP segment number, host buffer address, length)) to, for example, a particular memory location in a temporary buffer, an ULP buffer or an application buffer residing in the host buffer 60.

[48] In other embodiments according to the present invention, some of these steps can be performed substantially in parallel or in a different order. For example, if the ULPDU is aligned within a transport protocol segment, then the headers of the various processing layers (e.g., L2, L3, etc.) and the CRC may be easily located. The header information may be analyzed, at least in part, in parallel or in a different order. Thus, NIC architectures that include multiple processing layers may benefit substantially in configuration and in operation when the incoming transport protocol segments include aligned ULPDUs.

[49] In various embodiments according to the present invention, the arrangements in FIGS. 2-4 may accommodate flow-through NIC architectures or non-flow-through NIC architectures. In some embodiments according to the present invention, FIGS. 2-4 may accommodate aligned ULPDUs (e.g., aligned MPA FPDUs or other aligned protocol data units) or unaligned ULPDUs (e.g., unaligned MPA FPDUs or other unaligned protocol data units). In some embodiments according to the present invention, FIGS. 2-4 may

accommodate in-order transport protocol segments or out-of-order transport protocols segments.

[50] With respect to some embodiments according to the present invention, the above-described processing of aligned ULPDUs, unaligned ULPDUs, in-order transport protocol segments or out-of-order transport segments by a flow-through NIC architecture may be applied in part or in whole to a non-flow-through NIC. Furthermore, the above-described processing of aligned ULPDUs, unaligned ULPDUs, in-order transport protocol segments or out-of-order transport segments by a non-flow-through NIC architecture may be applied in part or in whole to a flow-through NIC.

[51] With respect to some embodiments according to the present invention, the processing of incoming frames from the network as set forth herein does not have to be accomplished in the order set forth herein. The present invention also contemplates processing incoming frames using a different order of processing steps. Moreover, the present invention also contemplates that some of the processing steps may be accomplished in parallel or in series.

[52] One or more embodiments according to the present invention may benefit from one or more advantages as set forth below.

[53] Substantial receiver optimizations may be achieved by implementing header alignment and carrying complete FPDUs. The optimizations allow for using substantially fewer buffers on the receiver system 30 (e.g., fewer buffers on a NIC of the network subsystem 80 or fewer buffers on a chipset 100 of the network subsystem 80) and fewer computations per FPDU. The optimizations may allow for the building of a flow-through receiver system 30 (e.g., a flow-through NIC of the network subsystem 80) that may enable TCP-based solutions to scale to 10Gbps and beyond. The optimizations may find use, for example, in hardware implementations of receiver systems 30 that process, in an expedited manner, multiple protocol layers such as, for example, L2 (e.g., Ethernet), TCP/IP and ULP (e.g., MPA/DDP) on top of TCP. The optimizations provide even greater efficiencies as the network speed increases, thereby accentuating the performance of a hardware-based receiver system.

[54] The alignment of one or more FPDUs in a TCP segment may provide greater flexibility with respect to the classification of an incoming TCP segment. For example, when the FPDUs are not aligned, the receiver system 30 may have to classify incoming traffic before it can calculate the FPDU CRC. However, if the FPDUs are aligned, then the operations order may be left to the discretion of the implementer.

[55] The alignment of one or more FPDUs in a TCP segment may substantially simplify the receiver algorithm. For example, there may be no need or a reduced need to locally buffer portions of FPDUs or to access state information to determine FPDU boundaries. There may be no need or a reduced need to access state information before a CRC calculation commences, thereby reducing internal latencies. There may be no need or a reduced need to have separate DMA accesses through the CRC machine or to have separate DMA activity for moving data to a buffer in the host memory 60.

[56] The alignment of one or more FPDUs in a TCP segment may provide efficiencies in processing in-order TCP segments and out-of-order TCP segments. For example, the receiver system 30 may use substantially the same mechanisms in either case. One of the few differences may occur, for example, in the accounting of the in-order TCP segments and the out-of-order TCP segments which may be handled separately. Header alignment and a guarantee that an integer number of complete FPDU in each TCP segment may result in the receiver system 30 performing direct data placement of out-of-order TCP segments with no need or a reduced need for buffering.

[57] The reduced need for buffering may make hardware implementations feasible in the form of a NIC of the network subsystem 80 in which buffering may be supported by on-board memory. In fact, the reduced need of buffering may make hardware implementations feasible in the form of a single integrated chip in which buffering may be supported by on-chip memory.

[58] The alignment of one or more FPDUs in a TCP segment may provide for receive buffers whose size does not scale with the number of connections. An aligned FPDU approach may be expected to scale more gracefully (i.e., less than linearly) as network speed

increases. Furthermore, if the system interface of a network controller offers ample bandwidth compared with the network bandwidth, then the aligned FPDU approach may allow buffer size to be substantially indifferent to network speed.

[59] While the present invention has been described with reference to certain embodiments, it will be understood by those skilled in the art that various changes may be made and equivalents may be substituted without departing from the scope of the present invention. In addition, many modifications may be made to adapt a particular situation or material to the teachings of the present invention without departing from its scope. Therefore, it is intended that the present invention not be limited to the particular embodiments disclosed, but that the present invention will include all embodiments falling within the scope of the appended claims.